Speicher und Tuning

Schlangenöl für den Speicher

Ausgabe: 03/2004
Seite: 12
Zeitschrift: Windows IT Pro »

RAM-Optimierungsprogramme mit falschen Versprechen

„Schlangenöl“ für den Speicher

Immer wieder tauchen Hilfsprogramme auf, die wahre Wunderdinge versprechen: Ein Windows-System, das nie mehr abstürzt oder eine Vergrößerung des effektiv zur Verfügung stehenden Hauptspeichers ohne Aufrüstung der Hardware. Unser Autor Mark Russinovich hat sich diese RAM-Optimierungsprogramme einmal näher betrachtet und fällt ein vernichtendes Urteil: Im besten Fall bewirken solche Programme nichts!

Wer öfter im World Wide Web unterwegs ist, wurde sicher schon mit Pop-Up-Fenstern konfrontiert, die mit großartigen Versprechungen für Produkte warben, die „den Hauptspeicher defragmentieren und die Gesamtleistung des Systems deutlich verbessern“ oder versprechen, „Anwendungs- und Systemfehler zu minimieren und mehr Hauptspeicher freizugeben“. Klickt man dann auf einer dieser Links, so wird man zu Werkzeugen und Hilfsprogrammen geführt, die diese Möglichkeiten und mehr für 10,- Euro, 14,95 Dollar oder 29,95 Euro versprechen – klingt eigentlich zu gut, um wirklich wahr zu sein. Und genauso ist es auch: Zwar scheinen diese Art von Programmen sehr nützlich zu sein und auch ordentliche Arbeit zu verrichten, doch im besten Fall bewirken diese „RAM-Optimierer“ nichts. Im schlimmsten Fall kann der Einsatz solcher Programme sogar dazu führen, dass der Gesamtdurchsatz eines Systems deutlich gesenkt wird!

Es sind sicher mehr als ein Dutzend solcher „Speicheroptimierer“ auf dem Markt erhältlich, wobei es hier sowohl kommerzielle als auch Freeware-Produkte gibt. Vielleicht haben viele Leser (wie auch wir in der Redaktion) solche Programme bereits auf ihren Systemen ausprobiert oder setzten sie immer noch ein. Doch was tun diese Art von Programmen wirklich im System, und wie schaffen sie es, den Eindruck zu erwecken, dass sie ihren selbst gewählten Ansprüchen wirklich gerecht werden? Dieser Artikel wirft einen Blick hinter die Kulissen solcher Optimierungsprogramme. Der Autor stellt dabei genau dar, wie die Software die entsprechenden sichtbaren Anzeigen für den Hauptspeicher unter Windows manipuliert.

Der erste Eindruck – das „Look & Feel“ der Oberfläche

Die Oberfläche solcher Programme zeigt zumeist eine grafische Übersicht an, in der neben dem aktuell zur Verfügung stehenden Hauptspeicher auch ein Grenzwert angezeigt wird, bei dem das Produkt seine Arbeit beginnen wird. Meistens ist bei dieser Darstellung auch noch eine weitere Markierung zu sehen, die den Memory-Bereich anzeigt, den das Werkzeug versuchen wird, zu „befreien“. Normalerweise ist es für den Anwender dieser Programme möglich, diese beiden Einstellungen nach seinen Wünschen zu konfigurieren. Auch eine manuelle Einstellung der Optimierung oder die regelmäßige automatische Optimierung des Hauptspeichers sind bei den meisten Programmen über ein Menü auswählbar. Einige Produkte ergänzen diese Informationen durch eine Anzeige der Prozesse, die auf dem jeweiligen System aktiv sind. Läuft ein Job, der von der Anwendung automatisch ausgeführt wird, so zeigt die Anzeige für den verfügbaren Speicherplatz häufig deutlich an – bei einigen Produkten nimmt der angezeigte freie Speicherplatz geradezu dramatisch zu. Damit soll natürlich der Eindruck erweckt werden, dass die Software nun Speicherplatz auf dem System frei geräumt hat, der für andere Anwendungen zur Verfügung steht. Um zu verstehen, wie es ein solches Programm

schafft, einen solch deutlichen Anstieg eines Werts zu erreichen, ist es zunächst einmal notwendig, sich zunächst einmal grundsätzlich mit der Art und Weise zu befassen, wie Windows den physikalisch zur Verfügung stehenden Speicher verwaltet.

Das „Gedächtnis“ des Rechners: Windows Memory Management

Genau wie bei anderen modernen Betriebssystemen kommt bei Windows ein so genannter virtueller Arbeitsspeicher zum Einsatz, der mit einem „Demand-Paging-System“ arbeitet. Ein Betriebssystem verwendet diesen virtuellen Speicher dazu, den Anwendungen „den Eindruck zu vermitteln“, dass ein weitaus größerer Hauptspeicher vorhanden sei, als er physikalisch im System installiert ist. Auf 32-Bit-Windows-Systemen steht für die Prozesse ein virtueller Adressraum von 4 GByte zur Verfügung. Dieser Adressraum wird vom Betriebssystem üblicherweise zu gleichen Teilen dem eigentlichen Prozess und dem System zugeteilt. Auf diese Weise kann ein Prozess bis zu 2 GByte des virtuellen Adressraums für sich belegen, abhängig davon, wie viel Speicher zur Verfügung steht. Die Gesamtsumme des allokierten virtuellen Speichers kann den Wert nicht überschreiten, der sich aus der Addition der Paging-Dateien und einem Großteil des vorhandenen physikalischen Speichers in dem entsprechenden System ergibt. Hier geht man nicht vom kompletten installierten Hauptspeicher des Rechners aus, weil sich das Betriebssystem standardmäßig einen kleinen Bereich des RAMs reserviert.

Geht man davon aus, dass Prozesse unter Zuhilfenahme einer entsprechend großen Paging-Datei in der Lage sind, virtuellen Speicher zu allokieren, dessen Größe die des installierten Hauptspeichers übersteigt, so hat das so genannte Memory-Manager-Subsystem des Windows-Rechners einige Aufgaben zu bewältigen: Es muss den vorhandenen physikalischen Speicher zwischen den Prozessen und den Daten aufteilen, die der Cache-Manager in seinem Cache-Speicher verwaltet. Die Skizze in Bild 1 zeigt, dass der Memory-Manager jedem Prozess, beispielsweise Microsoft Word, Notepad und dem Windows-Explorer, einen Teil des vorhandenen physikalischen Speichers zuordnet, dieser Bereich wird als das „Working Set“ des Prozesses bezeichnet. Die Teile des Betriebssystem-Kernels und der verschiedenen Treiber, die das System auslagern kann, werden „pageabel“ genannt. Ebenso wie die Pufferspeicher des Kernels, die als „paged pool“ bezeichnet werden, und der Bereich des physikalischen Speichers, der vom Cache-Manager verwaltet wird, bekommen all diese auslagerfähigen Seiten ihr eigenes „Working Set“ zugeteilt, das die Bezeichnung „System Working Set“ trägt.

Der Memory-Manager erweitert und verkleinert die „Working Sets“ der Prozesse und des Systems je nach Anforderung der Prozesse, die ja einen möglichst schnellen Zugriff sowohl auf ihren Programmcode als auch auf ihre Daten benötigen. Die Hardware, auf die der Memory-Manager aufsetzt, verlangt vom Windows-Betriebssystem, dass es die „Working Sets“ und den virtuellen Speicher in der Größe von Speicherseiten („Pages“) verwaltet. Auf einem 32-Bit-Prozessor der x86-Serie besitzen diese „Pages“ typischerweise eine Größe von 4096 Bytes. Trotzdem verwenden das Betriebssystem und sehr speicherintensive Anwendungen aus Optimierungsgründen nach Möglichkeit auch große Speicherseiten mit einer Größe von 4 MByte.

Versucht ein Prozess auf eine Speicherseite seines virtuellen Speichers zuzugreifen, die in seinem „Working Set“ nicht vorhanden ist, so wird eine Hardware-Exception ausgelöst, die als „page fault“ bekannt ist. Der Memory-Manager weist den Daten, auf die jetzt neu zugegriffen wird, eine Speicherseite aus dem Bereich des physikalischen Speichers zu, der zu diesem Zeitpunkt verfügbar ist. Zusätzlich kann der Memory-Manager jetzt auch die Entscheidung fällen, das „Working Set“ des entsprechenden Prozesses um diese Speicherseite zu erweitern. Sollte der Memory-Manager allerdings befinden, dass dieses „Working-Set“ bereits groß genug ist, wird eine Seite aus dem aktuellen „Working Set“ gegen die neue Speicherseite ausgetauscht. Dabei wird in der Regel die Seite ausgewechselt, auf die der Prozess am längsten nicht mehr zugegriffen hat. Dabei wird davon ausgegangen, dass der Prozess diesen Bereich in naher Zukunft nicht so schnell

wieder brauchen wird. Wenn der Memory-Manager eine Speicherseite aus dem „Working Set“ herausnimmt, steht die Entscheidung an, was mit dieser Seite dann zu tun ist. Wurde diese verändert, so wird der Memory-Manager sie zunächst auf die Liste der modifizierten Seiten (modified page list) setzen. In dieser Liste werden jene Speicherseiten vermerkt, die schließlich in die Paging-Datei oder in die „Memory-Mapped“-Dateien, die zu den Seiten gehören, geschrieben werden. Der Memory-Manager bewegt die Seiten von dieser Liste in einen Bereich, der als „standby list“ bezeichnet wird (Bild 2). Seiten, die nicht modifiziert wurden, „wandern“ direkt auf diese „Standby“-Liste. Deshalb kann man diesen Bereich auch als einen Cache-Speicher für die File-Daten bezeichnen.

Was dem System zur Verfügung steht: freier Speicherplatz

Es wurde zuvor erwähnt, dass der Memory-Manager einem Prozess, der einen „page fault“ bekommt, eine Seite aus dem freien, verfügbarem Speicher zuweist. Nun bleibt aber noch zu klären, wie „verfügbarer Speicher“ (availible memory) im System definiert ist. Die gerade beschriebene „Standby“-Liste stellt dabei einen Teil des physikalischen Hauptspeichers dar, den der Memory-Manager als „verfügbar“ betrachtet. Zum verfügbaren Speicher werden ebenfalls solche Speicherbereiche gerechnet, in denen sich Seiten mit Daten befinden, die zu einem Abschnitt des virtuellen Speichers gehören, der vom System bereits wieder freigegeben wurde (deallokiert). Dazu gehören beispielsweise Speicherseiten mit Daten von Prozessen, die bereits wieder beendet wurden. Weiterhin gehören solche „Pages“ dazu, die freigegeben wurden und anschließend von einem speziellen Thread des Memory-Managers mit „Nulldaten“ gefüllt wurden. Dieser Thread ist in niedriger Priorität aktiv und wird als „Zero Page Thread“ (Bild 2) bezeichnet. Diese Art von Seiten werden auf der Liste der freien Seiten (Free List) beziehungsweise auf der Liste der „Null-Seiten“ (Zeroed Page List) gespeichert.

Die Skizze in Bild 2 verdeutlicht die Vorgänge, die beim Übergang von den „Working Sets“ zur Liste der Speicherseiten (page list) ablaufen. Ein System-Thread wird einmal pro Sekunde vom Betriebssystem ausgeführt, dessen Aufgabe darin besteht, den Working-Set-Manager des Memory-Managers anzustoßen, der dann das System untersucht und die „Working Sets“ bearbeitet. Wenn der Bereich des verfügbaren Speichers klein geworden ist, wird diese Komponente den Prozessen Speicherseiten, die in den letzen Sekunden keine „page faults“ ausgelöst haben, entziehen und diese aus dem Speicher entfernen. Diese Seiten „wandern“ dann zur Liste der modifizierten Seiten (Modified Page List) oder zur „Standby“-Liste und erhöhen damit den Wert des zur Verfügung stehenden Speichers. Dieser „Tuning“-Mechanismus besitzt noch einen wichtigen Seiteneffekt für das System: Wenn das Betriebssystem für andere Prozesse Speicher benötigt, nimmt der Memory-Manager diesen Speicher aus den „Working Sets“ solcher Prozesse, die sich gerade im „idle“-Status befinden, also im Prinzip „untätig“ sind. Dadurch kann es passieren, dass diese „Working Sets“ schließlich komplett verschwinden. Das bedeutet wiederum, dass Prozesse, die über eine gewisse Zeit „idle“ sind, schließlich auch keinen Bereich des physikalischen Speichers mehr belegen.

Benötigt ein Prozess eine neue Speicherseite des physikalischen Speichers, so prüft der Memory-Manager zunächst, ob die Seite, auf die der Prozess zugreift, sich auf der „Standby“- oder der „Modified-Page“-Liste befindet. Eine Speicherseite wird sich auf einer dieser Listen befinden, wenn sie aus dem „Working Set“ entfernt wurde und in der Zwischenzeit nicht für andere Zwecke oder von anderen Prozessen benutzt wurde. Das „Zurückbringen“ einer solchen Seite in das „Working Set“ eines Prozesses wird auch als „soft page fault“ bezeichnet, da es im Gegensatz zum „hard page fault“ nicht das Lesen von Daten aus der Paging-Datei oder gar aus einer Datei von der Festplatte erfordert. Findet der Memory-Manager die gesuchte Seite weder in der „Standby“- noch in der „Modified Page“-Liste, so wählt er eine der anderen Listen aus, die eine Speicherseite zur Verfügung stellen. Dabei wird er zunächst die „Free List“, dann die „Zeroed Page List“ und schließlich die „Standby“-Liste nach einer solchen Seite absuchen. Sollte kein Speicher zur Verfügung stehen, so stößt der Memory-Manager den Balance-Set-Manager an, dessen Aufgabe unter anderem darin besteht, die „Working Sets“ so „zurecht zu

stutzen“, dass dann wieder freier Speicherplatz auf einer der drei Listen zur Verfügung steht. Wenn der Memory-Manager gezwungen ist, eine Speicherseite von einer der drei Listen zu entfernen, muss er zunächst die Entscheidung treffen, wie auf den entsprechenden Programm-Code oder die Daten zugegriffen werden soll. Dabei stehen ihm unter anderem die folgenden Möglichkeiten zur Verfügung: Die Daten können aus der Paging-Datei oder einem ausführbaren Image gelesen werden. Natürlich kann zu diesem Zeitpunkt auch der Fall eintreten, dass mit dem Wert „0“ aufgefüllte Daten (zero-filled data) angelegt werden müssen: Das ist immer dann notwendig, wenn die entsprechende Anwendung einen neuen „frischen“ Datenbereich anlegen will und die entsprechende Speicherseite nicht aus der „Zeroed Page“-Liste stammt.

Die wundersame Vermehrung – oder – die Erschaffung von verfügbarem Speicherplatz

Mit diesem Wissen über die grundsätzliche Arbeitsweise des Memory-Managers unter Windows können wir uns nun wieder den RAM-Optimierungsprogrammen zuwenden. Der Wert, der von dieser Art Programmen als verfügbarer Speicher angezeigt wird, entspricht exakt dem Wert, den der Windows-Task-Manager anzeigt (Bild 3). Hier werden sowohl der physikalisch im System vorhandene als auch der im Augenblick zur Verfügung stehende Speicher unter „Systemleistung“ angezeigt. Dieser Wert, der im Bild 3 ungefähr 700 MByte umfasst, stellt die Summe der drei Listen „Standby“, „Zeroed Page“ und „Free List“ dar, während der Wert darunter, der mit „System-Cache“ bezeichnet ist und auf dem Testsystem zirka 400 MByte umfasst, sich aus den Werten des „System Working Set“ und der „Standby“-Liste zusammensetzt. Unter Windows NT 4.0 und allen früheren Windows-Versionen entspricht der Wert Datei-Cache (File Cache) nur der Größe des „System Working Set“.

Die RAM-Optimierer nutzen aus, wie der Memory-Manager arbeitet: Sie allokieren zunächst große Mengen des virtuellen Speichers, um diesen dann später wieder insgesamt freizugeben. Die Skizzen in Bild 4 zeigen die Effekte, die ein solches RAM-Optimierungsprogramm auf das System hat. Der obere Teil der Skizze zeigt den Zustand der „Working Sets“ und des verfügbaren Speichers vor der Optimierung. Die Darstellung darunter verdeutlich das weitere Vorgehen dieser Software: Sie verursacht zunächst einmal einen sehr hohen Speicherbedarf, was durch das Auslösen vieler „page faults“ in kurzer Zeit erreicht wird. Als Antwort darauf wird der Memory-Manager das „Working Set“ des Optimierers vergrößern. Diese Vergrößerung geht dabei aber zu Lasten des verfügbaren freien Arbeitsspeichers. Sollte dieser auch „aufgebraucht“ sein, so werden die „Working Sets“ anderer Programme verkleinert. Die dritte Skizze in Bild 4 verdeutlicht diesen Zustand: Nachdem das RAM-Optimierungsprogramm seinen Speicherbereich freigegeben hat, verschiebt der Memory-Manager alle Speicherseiten, die bisher dem Optimierer zugeordnet waren, auf die „Free List“, womit sie zum verfügbaren freien Speicherplatz hinzugerechnet werden. Die meisten der Optimierungsprogramme verstecken den rapiden Abfall des verfügbaren Speichers, der während des ersten Schritts logischerweise auftreten muss. Wenn man allerdings den Task-Manager des Windows-Systems während des Optimierungsvorgangs startet, so kann dieser starke Abfall bei den Werten für den verfügbaren Speicher in der Regel beobachtet werden.

Auf den ersten Blick mag es nun so erscheinen, als habe man durch dieses Vorgehen doch genau das erreicht, was die entsprechenden Programme versprechen: Es steht mehr freier Speicher zur Verfügung – aber dieser Eindruck ist falsch! Wenn Speicheroptimierer die Anzeige des verfügbaren Speichers nach oben treiben, dann zwingen sie den Programm-Code und die Daten anderer Programme dazu, Speicherbereiche abzugeben und ganz aus dem Speicher zu „verschwinden“. So könnte beispielsweise auf einem solchen System zum Zeitpunkt der Optimierung gerade Microsoft Word ausgeführt werden. Wenn der Optimierer nun mehr verfügbaren Speicher „freischaufelt“, müssen sowohl der Programm-Code als auch der Text eines in Word offenen Dokuments, die bisher Teil des „Working Sets“ von Microsoft Word waren und sich somit im physikalischen Speicher befanden, wieder von der Festplatte gelesen

werden, wenn der Anwender seiner Text bearbeitet. Der Geschwindigkeitsverlust kann sich in solch einem Fall besonders auf Serversystemen stark auswirken, weil die Daten, die in der „Standby“-Liste und im „Working Set“ des Systems gecached wurden, nun bereits wieder verworfen sein können, sich also nicht mehr im Speicher befinden. Das gilt dann natürlich auch für den Programm-Code und die Daten von zu diesem Zeitpunkt aktiven Serveranwendungen.

Was noch versprochen und nicht gehalten wird

Einige Anbieter solcher Optimierungsprogramme machen aber noch ganze andere Versprechungen: So wird häufig behauptet, die Produkte seien in der Lage, Speicherbereich freizugeben, die unnötigerweise von nicht aktiven Prozessen belegt werden. Damit werden dann häufig solche Programme gemeint, die sich beispielsweise im „Taskbar Tray“ befinden. Diese Behauptung ist schlichtweg falsch, da das Windows-System die „Working Sets“ der „idle“-Prozesse ganz automatisch zurechtstutzt. Der Memory-Manager bewältigt also bereits standardmäßig diese Art der Optimierung. Die Entwickler der Speicheroptimierer weisen zudem immer wieder darauf hin, dass ihre Werkzeuge in der Lage sein sollen, den Hauptspeicher zu defragmentieren. Wenn ein großer des virtueller Speichers allokiert und anschließend wieder freigegeben wird, so kann ein denkbarer Seiteneffekt dieser Operationen darin bestehen, dass Blöcke von zusammenhängendem verfügbaren Speicher entstehen. Das Funktionsprinzip des virtuellen Speichers besteht aber unter anderem darin, dass die genaue Aufteilung des physikalischen Speichers gegenüber den Prozessen verborgen beziehungsweise maskiert wird. Deshalb können die Prozesse auch keinen Vorteil aus der Tatsache ziehen, dass der virtuelle Speicher direkt auf zusammenhängendem physikalischen Speicher aufsetzt. Während Prozesse im System ausgeführt werden und deren „Working Sets“ dabei vergrößert oder verkleinert werden, wird die Zuordnung von virtuellem zu physikalischem Speicher auf jeden Fall fragmentiert werden, ganz gleich ob sich zusammenliegender Speicher dahinter befindet oder nicht

Allerdings existiert wirklich ein Fall, in dem das Vorhandensein von zusammenhängendem Speicher sich wirklich positiv auf die Geschwindigkeit auswirken kann: Der Memory-Manager kann eine Methode eines Mechanismen verwenden, der mit „page coloring“ bezeichnet wird. Diese Methode, mit deren Hilfe entschieden wird, welche Speicherseite aus der „Free“- oder der „Zeroed Page“-Liste einem Prozess zugeordnet wird, wird vom System verwendet, um das Verhalten der Speicher-Caches der CPU zu verbessern. Trotzdem wird jeder kleine Vorteil, der durch die Verfügbarkeit von physikalischem Speicher in zusammenhängender Form entstehen mag, durch die negative Effekte, die durch das „Herauswerfen“ von wichtigem Programm-Code und Daten entstehen, doch deutlich überwiegen. Schließlich weisen die Anbieter der Software häufig noch darauf hin, dass ihre Lösungen in der Lage seien, Speicherbereiche wieder zur Verfügung zu stellen, die durch so genannte „Speicherlecks“ (memory leaks) verloren gegangen seien. Diese Aussage ist sicher die Behauptung, die am meisten in die Irre führt. Der Memory-Manager „weiß“ zu jedem Zeitpunkt, welcher physikalische und virtuelle Speicherbereich zu welchem Prozess gehört. Wenn ein Prozess Speicher belegt und diesen aufgrund eines Programmfehlers nicht wieder freigibt, so wird dieses Verhalten allgemein als Speicherleck oder „Memory Leak“ bezeichnet. Der Memory-Manager ist bei so einem Fall allerdings nicht in der Lage festzustellen, dass der Prozess nicht mehr auf diesen allokierte Speicherbereich zugreift und muss warten bis der Prozess beendet ist, um dann den Speicher wieder „aufzuräumen“. Sogar wenn ein Prozess, der ein solches Speicherleck verursacht, nicht wieder automatisch beendet wird, wird der Memory-Manager das „Working Set“ des entsprechenden Prozesses nach und nach um die Speicherseiten verkleinern, auf die aufgrund des Programmierfehlers nicht mehr zugegriffen wird. Diese Seiten werden – genau wie andere Speicherseiten auch – in die Paging-Datei ausgelagert, und der physikalische Speicher steht wieder anderen Anwendungen zur Verfügung. Auf diese Weise wird ein solches Speicherleck nur einen sehr geringen Einfluss auf den zur Verfügung stehenden freien physikalischen Speicher haben. Der wirkliche Einfluss eines solchen Fehlers wird sich nur beim virtuellen Speicher bemerkbar machen, ein Wert, der im Task-Manager sowohl in der Grafik

„Auslagerungsdatei“ und in dem Wert „Zugesicherter Speicher“ dargestellt ist (Bild 3). (fms)